UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/730,576	12/08/2003	Frank S. Filz	BEA9-2003-0016-US1	3051

49056 7590 12/12/2008
LIEBERMAN & BRANDSDORFER, LLC
802 STILL CREEK LANE
GAITHERSBURG, MD 20878

EXAMINER	
JOHNSON, JOHNESE T	

ART UNIT	PAPER NUMBER
2166	

MAIL DATE	DELIVERY MODE
12/12/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	**Application No.** 10/730,576	**Applicant(s)** FILZ, FRANK S.	
	Examiner Johnese Johnson	**Art Unit** 2166	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *22 October 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,3-6,8-12,14,16-19 and 21-25* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1, 3-6, 8-12, 14, 16-19, and 21-25* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Continued Examination Under 37 CFR 1.114

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on October 22, 2008 has been entered.

Remarks

2. In response to the Amendment filed on October 22, 2008, claims 1, 3-6, 8-12, 14, 16-19, and 21-25 are pending.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1, 3-6, 8-12,14,16-19 and 21-25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Short et al. (US Pat. No. 6,178,529), in view of Szabo et al.

(US Pat. No. 7,065,746), Frank et al.(U.S. Pat. No. 6,871,222), and further in view of Engel (US Pat. No. 6,681,389).

As to claim 1, Short et al. disclose:

creating a version control system including a disk header record of a shared storage resource (as Short discloses every disk in the RAID has a header file which is the first file that is read when the disk is read - see col. 4, lines 30-31**) and a version control record within said shared resource, said version control record comprising all versions of each type of data structure in said shared resource (**as Short discloses a version control mechanism in the startup function that contains all versions of the software in the shared resource. see col. 9, lines 20-22**).**

However, Short et al. does not explicitly disclose:

said version control record to organize meta data in a known location in said shared resource in communication with said cluster; and

validating software compatibility of a new cluster member with target data retained in said shared resource assigned to the cluster separately using the disk header record and the version control record prior to a new cluster member joining said cluster.

Szabo et al. discloses:

said version control record to organize meta data in a known location in a shared resource in communication with said cluster (as Szabo discloses a version control mechanism that organizes data about changes in versions. see Szabo et al., col. 5, lines 17-38**).**

It would have been obvious to have modified the teachings of Short et al. by the teachings of Szabo et al. to provide a computerized method and system of managing the integrity of an integrated applications environment because a highly integrated system can create interdependencies where a small change in one application may adversely impact obvious or seemingly unrelated applications. Each change can cause one or more component that depends on the changed application to become unstable, thereby compromising the integrity of the integration **(**see Szabo et al. col. 1. lines 53-55 and Szabo et al. col. 1, lines 23-26, 29-32**).**

However, the combination of Short et al. and Szabo et al. do not disclose:

validating software compatibility of a new cluster member with target data retained in said shared resource assigned to the cluster separately using the disk header record and the version control record prior to a new cluster member joining said cluster; and, said new cluster member joining said cluster responsive to validation of software compatibility.

Frank et al. disclose:

validating software compatibility of a new cluster member with target data retained in said shared resource assigned to the cluster separately using the disk header record (as Frank et al. discloses software validation using the disk header record. see col. 4, lines 1-15**) and the version control record prior to a new cluster member joining said cluster (**as Frank et al. disclose software validation using a version control record before the node is allowed to join the cluster. see col. 6, lines 23-36**); and,**

said new cluster member joining said cluster responsive to validation of software compatibility (as Frank et al. discloses the node joining the cluster after confirmation that the version is version 2. see col. 4, lines 1-15**).**

It would have been obvious to have modified the teachings of Short et al. and Szabo et al. by the teachings of Frank et al. to determine if a node was compatible with the cluster members and had access to data; and, to preserve the integrity of the shared data **(**see Frank et al. col. 2, lines 7-22**).**

However, Short et al., Szabo et al, and Frank et al. do not explicitly disclose:

the date structure in the shared resource representing target data of a software application;

Engel discloses:

the date structure in the shared resource representing target data of a software application (see col. 3, lines 56-64)**;**

It would have been obvious to have modified the teachings of Short et al. Szabo et al., and Frank et al., by the teachings of Engel to update platform controlling or cluster controlling software as well as application software on all operating machines/servers in a cluster without manually taking each machine/server offline and performing a software update installation (see Engel, col. 1, lines 31-40).

As to claim 3, Short et al., as modified, disclose:

wherein the step of validating software compatibility of a new cluster member includes scanning said version control record for said data structure version conflict (as Short discloses checking the version control mechanism for the correct version number. see Short et al., col. 9, lines 15-22**).**

As to claim 4, Short et al., as modified, disclose:

maintaining a table within said version control record of an operating software version of each node in said cluster (as Short discloses a table comprising all supported versions. see Short et al., col. 9, lines 33-35**).**

As to claim 5, Short et al., as modified, disclose:

validating compatibility of each node in said cluster with said operating software version table (as Short discloses software validation using a version table. see Short et al., col. 9, lines 15-22, 33-35 and Short et al., col. 6, lines 59-62**) prior to upgrading each data structure in said shared resource (**as Short discloses a global update. see Short et al., col. 5, lines 36-38 **).**

As to claim 6, Short et al., as modified, disclose:

wherein the step of validating compatibility of each of said nodes in said cluster is inclusive of inactive cluster nodes (as Short discloses nodes that are inactive and are trying to join the cluster. see Short et al., col. 5, lines 12-21**).**

As to claim 8, Short et al. disclose:

at least two nodes to operate in a computer cluster, each of said nodes having a processor and memory, and in communication with a storage network (as Short discloses members of a cluster and as cited above each node has a processor and memory and in communication with a storage network. see col. 4, line 40**);**

a version control system in communication with said nodes, said version control system having a disk header and a version control record of a shared storage resource, (as Short discloses the Startup mechanism that functions as the version control record and . see col. 9, lines 33-35 and as Short discloses every disk in the RAID has a header file which is the first file that is read when the disk is read - see col. 4, lines 30-31**); and**

a version control record within said shared resource (as Short discloses the Startup mechanism that functions as the version control record. see col. 9, lines 20-22**),**

said version control record inclusive of all versions of each type of data structure in said shared resource (as Short discloses a version control mechanism in the startup function that contains all versions of the software in the shared resource. see col. 9, lines 20-22 and col. 9, lines 33-35**).**

However, Short et al. does not explicitly disclose:

said version control record to organize meta data in a known location in a shared resource in communication with said cluster.

a membership manager in communication with said version control system to validate compatibility of a new cluster member with target data retained within

each of said data structure with use of said version control record prior to acceptance of said new cluster member.

Szabo et al. discloses:

said version control record to organize meta data in a known location in a shared resource in communication with said cluster (as Szabo discloses a version control mechanism that organizes data about changes in versions. see col. 5, lines 17-38**).**

It would have been obvious to have modified the teachings of Short et al. by the teachings of Szabo et al. to provide a computerized method and system of managing the integrity of an integrated applications environment because a highly integrated system can create interdependencies where a small change in one application may adversely impact obvious or seemingly unrelated applications. Each change can cause one or more component that depends on the changed application to become unstable, thereby compromising the integrity of the integration **(**see Szabo et al. col. 1. lines 53-55 and Szabo et al. col. 1, lines 23-26, 29-32**).**

validating software compatibility of a new cluster member with storage media in said shared resource assigned to the cluster separately using the disk header record and the version control record prior to a new cluster member joining said cluster.

However, the combination of Short et al. and Szabo et al. do not disclose:

a membership manager in communication with said version control system to validate compatibility of a new cluster member with target data retained within

each of said data structure with use of said version control record prior to acceptance of said new cluster member.

Frank et al. disclose:

a membership manager in communication with said version control system to validate compatibility of a new cluster member with target data retained each of said data structure with use of said disk header record (as Short discloses membership managed by individual nodes and discloses a cluster manager. see col. 4, lines 1-24**) and said version control record prior to acceptance of said new cluster member (**as Short discloses a repository which includes records comprising versions. see col. 6, lines 23-36**).**

It would have been obvious to have modified the teachings of Short et al. and Szabo et al. by the teachings of Frank et al. to determine if a node was compatible with the cluster members and had access to data; and, to preserve the integrity of the shared data **(**see Frank et al. col. 2, lines 7-22**).**

However, Short et al., Szabo et al, and Frank et al. do not explicitly disclose:

the date structure in the shared resource representing target data of a software application;

Engel discloses:

the date structure in the shared resource representing target data of a software application (see col. 3, lines 56-64)**;**

It would have been obvious to have modified the teachings of Short et al. Szabo et al., and Frank et al., by the teachings of Engel to update platform controlling or cluster controlling software as well as application software on all operating machines/servers in a cluster without manually taking each machine/server offline and performing a software update installation (see Engel, col. 1, lines 31-40).

As to claim 9, Short et al., as modified, disclose:

an operating software version table within said version control record (as Short discloses a version control mechanism in the startup function that contains all versions of the software in the shared resource. see Short et al., col. 9, lines 33-35**).**

As to claim 10, Short et al., as modified, disclose:

a validation manager to validate compatibility of an existing cluster member with said operating software version table (as Short discloses individual nodes managing validation. see Short et al., col. 9, lines 15-22, 33-35 and Short et al., col. 6, lines 59-62**) prior to an upgrade of each data structure in said shared storage (**as Short discloses global updates. see Short et al., col. 5, lines 36-38 – global update**).**

As to claim 11, Short et al., as modified, disclose:

wherein said validation manager is inclusive of inactive cluster nodes (as Short discloses nodes having the same view of cluster membership and a regroup is also disclosed. see Short et al., col. 5, lines 12-22**).**

As to claim 12, Short et al., as modified, disclose:

a version manager to scan a data structure type record within said shared resource prior to access of said version control record by a cluster member (as Short discloses the startup function is checked. see Short et al., col. 5, lines 12-21**).**

As to claim 14, Short et al. disclose:

A computer-readable recordable storage medium;

instructions to provide a version control record system including a disk header record of a shared resource (see col. 4, lines 30-31 - every disk in the RAID has a header file which is the first file that is read when the disk is read**) and a version control record of a shared resource, said version control record inclusive of each type of data structure in said shared resource; (**as Short discloses a version control mechanism in the startup function that contains all versions of the software in the shared resource. see col. 9, lines 33-35**).**

However, Short et al. do not explicitly disclose:

a version control record, said version control record to organize meta data in a known location in a shared resource;

instructions to validate compatibility of a new cluster member with target data retained within storage media in said shared resource assigned to a cluster using said version control record prior to said new cluster member joining said cluster.

Szabo et al. discloses:

a version control record, said version control record to organize meta data in a known location in said shared resource (as Szabo discloses a version control mechanism that organizes data about changes in versions. see col. 5, lines 17-38**).**

It would have been obvious to have modified the teachings of Short et al. by the teachings of Szabo et al. to provide a computerized method and system of managing the integrity of an integrated applications environment because a highly integrated system can create interdependencies where a small change in one application may adversely impact obvious or seemingly unrelated applications. Each change can cause one or more component that depends on the changed application to become unstable, thereby compromising the integrity of the integration **(**see Szabo et al. col. 1. lines 53-55 and Szabo et al. col. 1, lines 23-26, 29-32**).**

However, the combination of Short et al. and Szabo et al. do not disclose:

instructions to validate compatibility of a new cluster member with target data retained within storage media in said shared resource assigned to a cluster using said version control record prior to said new cluster member joining said cluster

Frank et al. disclose:

Instructions (see col. 11, lines 1-2**) to validate compatibility of a new cluster member with target data retained within storage media in said shared resource assigned to a cluster separately using said disk header record (**as Frank et al. discloses software validation using the disk header record. see col. 4, lines 1-15**) and said version control record prior to said new cluster member joining said cluster**

(as Frank et al. disclose software validation using a version control record before the node is allowed to join the cluster. see col. 6, lines 23-36**).**

It would have been obvious to have modified the teachings of Short et al. and Szabo et al. by the teachings of Frank et al. to determine if a node was compatible with the cluster members and had access to data; and, to preserve the integrity of the shared data **(** see Frank et al. col. 2, lines 7-22**).**

However, Short et al., Szabo et al, and Frank et al. do not explicitly disclose:

the date structure in the shared resource representing target data of a software application;

Engel discloses:

the date structure in the shared resource representing target data of a software application (see col. 3, lines 56-64)**;**

It would have been obvious to have modified the teachings of Short et al. Szabo et al., and Frank et al., by the teachings of Engel to update platform controlling or cluster controlling software as well as application software on all operating machines/servers in a cluster without manually taking each machine/server offline and performing a software update installation (see Engel, col. 1, lines 31-40).

As to claim 16, Short et al., as modified, disclose:

further comprising instructions to validate compatibility of each cluster member (as Short discloses software validation using a version table. see Short et al., col. 9, lines 15-22, 33-35 and Short et al., col. 6, lines 59-62**) prior to upgrading each data**

structure in said shared resource (as Short discloses a global update. see Short et al., col. 5, lines 36-38**).**

As to claim 17, Short et al., as modified, disclose:

wherein said compatibility validation instruction accesses an operating software version table within said version control record (as Short discloses accessing a table comprising all supported versions via the Startup function. see Short et al., col. 9, lines 33-35**).**

As to claim 18, Short et al., as modified, disclose:

wherein said validation instruction includes inactive cluster nodes (as Short discloses a regroup event. see Short et al., col. 5, lines 12-21**).**

As to claim 19, Short et al., as modified, disclose:

Instructions to scan a data structure type record prior to access of said version control record (as Short discloses the startup function is checked. see Short et al., col. 5, lines 12-21**).**

As to claim 21, Short et al., as modified, disclose:

wherein the step of validating software compatibility of said new cluster member with storage media (as Short discloses software validation. see Short et al., col. 5, lines 12-21**) includes determining said header record of a master disk in said**

shared resource (see Short et al., col. 4, lines 30-31 - every disk in the RAID has a header file which is the first file that is read when the disk is read**) is compatible with software operating in the new cluster member (**as Short discloses confirmation of node(s) having version 2. see Short et al., col. 9, lines 15-22, 33-35 and Short et al., col. 6, lines 59-62**).**

As to claim 22, Short et al., as modified, disclose:

wherein said membership manager (as Short discloses a membership manager. see col. 4, lines 63-64**) determines said header record of a master disk in said shared resource (**see Short et al., col. 4, lines 30-31 - every disk in the RAID has a header file which is the first file that is read when the disk is read**) is compatible with software operating in the new cluster member (**as Short discloses software validation. see Short et al., col. 9, lines 15-22, 33-35 and Short et al., col. 6, lines 59-62**).**

As to claim 23, Short et al., as modified, disclose:

wherein the instructions (as Short discloses instructions. see Short et al., col. 2, line 41**) to validating software compatibility of said new cluster member with storage media (**as Short discloses software validation during a regroup. see col. 5, lines 12-21**) includes instructions to said header record of a master disk in said shared resource (**see Short et al., col. 4, lines 30-31 - every disk in the RAID has a header file which is the first file that is read when the disk is read**) is compatible with software operating in the new cluster member (**as Short discloses confirmation of node(s) having version 2. see Short et al., col. 9, lines 15-22, 33-35 and Short et al., col. 6, lines 59-62**).**

As to claim 25, Short et al., does not explicitly disclose:

wherein said version control record is available to all server nodes that are members of the cluster and nodes requesting membership in the cluster.

However, discloses:

wherein said version control record is available to all server nodes that are members of the cluster and nodes requesting membership in the cluster (as Frank teaches the version control mechanism is available to nodes trying to join the cluster. see col. 9, lines 22-34).

It would have been obvious to have modified the teachings of Short et al. and Szabo et al. by the teachings of Frank et al. to determine if a node was compatible with the cluster members and had access to data; and, to preserve the integrity of the shared data **(** see Frank et al. col. 2, lines 7-22**).**

5. Claims 1, 3-6, 8-12,14,16-19 and 21-25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Short et al. (US Pat. No. 6,178,529), in view of Szabo et al. (US Pat. No. 7,065,746) in view of Frank et al.(U.S. Pat. No. 6,871,222), Thomas et al. (U.S. Pat. No. 6,460,052), and further in view of Engel (US Pat. No. 6,681,389).

As to claim 24, Short et al., as modified, does not explicitly disclose:

wherein each data structure is permanently assigned a position within the version control record.

However, Thomas discloses:

wherein each data structure is permanently assigned a position within the version control record (as Thomas teaches a repository used by the version control mechanism that serves as a persistent store for the objects managed by the version control mechanism, i.e. the objects are persistently or permanently stored in the version control repository in a table/ record. see col. 4, lines 19-30)**.**

It would have been obvious to have modified the teachings of Short et al., Szabo et al. , Frank et al. and Thomas, by the teachings of Engel to provide a configuration management system in which versioning of software components can be performed at finer and coarser granularities than is provided by the conventional file-based versioning systems (see Thomas col. 2, line 50-53).

Response to Arguments

6. Applicant's arguments with respect to claims 1-12, 14, 16-19, and 21-25 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

7. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Johnese Johnson whose telephone number is 571-270-1097. The examiner can normally be reached on 4/5/9.

If attempts to reach the examiner by telephone are unsuccessful, the examiner’s supervisor, Hosain Alam can be reached on 571-272-3978. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/J. J./
Examiner, Art Unit 2166

December 1, 2008
JJ

/Khanh B. Pham/

Primary Examiner, Art Unit 2166